

# Exploring Design Tradeoffs Of A Distributed Algorithm For Cosmic Ray Event Detection

Suhail Yousaf<sup>\*1</sup>, Rena Bakhshi<sup>1</sup>, Maarten van Steen<sup>1</sup>, Spyros Voulgaris<sup>1</sup>, and John L. Kelley<sup>2</sup>

<sup>1</sup>The Network Institute, and Dept. of Computer Science, VU University Amsterdam, De Boelelaan 1081, 1081 HV Amsterdam, The Netherlands

<sup>2</sup>Radboud University Nijmegen, IMAPP / Department of Astrophysics, Heyendaalseweg 135, 6500 GL Nijmegen, The Netherlands

## Abstract

Many sensor networks, including large particle detector arrays measuring high-energy cosmic-ray air showers, traditionally rely on centralised trigger algorithms to find spatial and temporal coincidences of individual nodes. Such schemes suffer from scalability problems, especially if the nodes communicate wirelessly or have bandwidth limitations. However, nodes which instead communicate with each other can, in principle, use a distributed algorithm to find coincident events themselves without communication with a central node. We present such an algorithm and consider various design tradeoffs involved, in the context of a potential trigger for the Auger Engineering Radio Array (AERA).

*Keywords:* distributed algorithm, sensor network, collaborative data analysis, cosmic ray

## 1 Introduction

There is an increasing trend in monitoring large spatial areas, such as forests for fire detection, mountains and hills for landslide and avalanches, and volcanoes for early warning or scientific study. This class of detectors also includes experiments measuring cosmic rays, high energy particles from space whose origin and acceleration mechanisms are still not understood. For example, the Pierre Auger Observatory consists of 1600 cosmic-ray detector nodes spread over an area of around 3000 km<sup>2</sup>. Similarly, LOFAR (Low Frequency ARray) for radio astronomy contains 8000 sensors and is spread over an area of more than 100 km in diameter. All these systems belong to the class of spatial sensor networks.

A characteristic feature of many of these spatial sensor networks is that each sensor node in the network collects data and sends it to a central unit for further analysis. However, as the amount of data that needs to be processed grows, and the area becomes larger, realizing the communication path from sensor to central unit becomes problematic. Firstly, the bandwidth requirements can be met only with special resources like

---

<sup>\*</sup>s.yousaf@vu.nl

fibre optic links. Installing a fibre optic or any other wired infrastructure is often infeasible (certainly in large areas). Consequently, wireless communication remains the only viable option. However, wireless communication has its own drawbacks, notably, limited bandwidth and unreliability of communication. Secondly, the central unit can easily become a bottleneck when the *number* of nodes grow and the *rate* at which data is collected grows.

Considering that sensed data often contains lots of raw measurements that will eventually be aggregated into much more informative units requiring much less space, local processing by nodes is essential for scalability. In many cases, significant improvements can be made if nodes in each other's proximity collaborate in the data analysis. Collaborative local data analysis may result in sending truly relevant aggregated (and location based) information to a central unit for further analysis, and may be the only path toward scalable solutions.

Local data analysis imposes additional requirements. For example, there may be a need for intensive communication with neighbouring nodes to reach a decision, leading to relatively high bandwidth usage. Likewise, while awaiting results from neighbours, temporary storage requirements may be fairly high unless special measures are taken. Finally, we need to take into account that local analysis may be computationally so demanding that algorithms may need to be tuned to what sensor nodes can realistically accomplish.

As it turns out, there are many tradeoffs to consider in building scalable solutions based on collaborative local data analysis. For example, how often and when should nodes exchange information? A low frequency of data exchange may lead to highly bursty traffic that may exceed local bandwidth constraints and that incur local buffering demands — which can be resolved by increasing the frequency of exchanges. However, frequent exchanges incur much higher energy costs caused by communication.

In this paper, we consider a specific, challenging application to illustrate the exploration of the design space for collaborative local data analysis in spatial sensor networks. The application involves the detection of ultra-high energy cosmic rays using a wireless sensor network. The application demands high communication bandwidth and involves large amounts of in-network data processing. On the other hand, the sensor nodes are resource constrained in terms of energy budget and capacity regarding computation and storage. In [1], we presented a distributed event detection algorithm which is entirely based on collaborative local data analysis. We explored the application-level resource requirements such as communication bandwidth for a certain level of performance in an unreliable communication environment.

We make two contributions. First, in contrast to the work described in [1], we explore in this paper the spectrum of tradeoffs that need to be considered while building scalable solutions based on collaborative local data analysis. The application that we consider is a natural fit for collaborative local data analysis. Therefore, a distributed system concept to detect high-energy cosmic-rays was proposed in [2]. However, the concept was not studied in further details. This is the first paper to our knowledge to explore the possibility of applying collaborative local data analysis in large-scale spatial wireless sensor networks to detect ultra-high energy cosmic rays. Second, we provide all the necessary details required to come to full understanding of our distributed event detection algorithm.

The rest of this paper is organized as follows. Section 2 presents our system model, specifying the assumptions made and semantics of events continuously occurring in our system. In Section 3 we describe our distributed event detection algorithm in detail. We illustrate, in Section 4, the tradeoffs to consider while building scalable solutions

based on collaborative local data analysis. In Section 5 we provide an experimental evaluation of our proposed distributed algorithm based on simulations. In Section 6 we discuss related work. Finally, in Section 7, we conclude the discussion and present our future work.

## 2 The System Model

Cosmic rays are high-energy charged particles, *e.g.* protons or atomic nuclei, that may originate from astrophysical objects such as supernova remnants and active galactic nuclei. Despite over a century of study, however, their origin and acceleration process remain a mystery.

At the highest energies, the flux of cosmic rays decreases to 1 particle per square kilometer per century, making direct detection infeasible. Instead, large spatial areas are instrumented with particle detectors that then measure the *extensive air shower* created when the cosmic ray interacts with the atmosphere of the Earth. In this work, we refer to the spatial area hit by a cosmic-ray air shower as the *event region*.

Multiple techniques can be used to detect cosmic-ray air showers. The Pierre Auger Observatory in Argentina uses a hybrid array of water-Cherenkov particle detectors and fluorescence telescopes to record the shower and indirectly measure the direction, energy, and composition of the original cosmic-ray [3]. Additionally, an enhancement is being deployed to detect the radio emission from the air showers: AERA [4], the Auger Engineering Radio Array. Because of the demands of the radio-detection method in particular (see Section 2.4), AERA is particularly suited for consideration as a testbed for collaborative local data analysis. We describe the AERA antenna array, and its triggering scheme, as a model for these studies, while noting that the distributed event-detection scheme can also be applied to other large-scale cosmic-ray-air-shower experiments.

### 2.1 System Setup

We consider a vast field covered by a large collection of antenna stations. Each antenna station – from now on called a **station** is a wireless sensor. The station can sense radio signals (in a specific frequency range) and can communicate with neighbouring stations in the field through a low-power wireless medium. Each station is attached with a standalone energy-harvesting device (*e.g.*, a solar panel), implying a modest energy budget per station. Each station has limited processing capabilities and a storage capacity in the order of a few hundred megabytes. The latter may seem much, but the incoming stream of raw, digitized samples that need to be analysed for cosmic rays is such that storage is quickly filled up. As a consequence, data analysis needs to be done within a limited time in order to free storage space.

We assume that a GPS receiver is attached to each station allowing (1) the clocks of stations to be globally synchronized,<sup>1</sup> and (2) stations to be aware of their location. The stations are assumed to be stationary. Each station is capable of communicating with at least one other station in the field. Furthermore, the system has been set up in such a way that under normal conditions the network of sensor nodes is strongly connected. In this paper, to illustrate the basic of our solution, we assume that all the communication channels are reliable.

---

<sup>1</sup>The accuracy is maintained within 10 nanoseconds through special devices

Each station relays its data to a base station called the Central Radio Station (CRS). The CRS has comparatively high capacity and sufficient energy. Among different possibilities of placement of stations in the field are a *grid-based* placement (e.g, triangular, rectangular, or hexagonal etc.) and a *uniform random* placement. We assume a grid-based placement.

## 2.2 Neighbourhood Semantics

We consider two notions of neighbourhood of a station:

1. **Geographical neighbourhood:** The geographical neighbourhood of a station is defined as the set of stations within a distance  $D$  from the station. We assume that a station has knowledge about its geographical neighbours. The geographical neighbourhood of a station  $p$  may change due to addition, removal, or failure of one or more stations within distance  $D$  from  $p$ . In this paper, we assume there is no change in geographical neighbourhood.
2. **Network-level neighbourhood:** In this case, the set of stations is represented by a graph  $G(V, E)$ , where vertex set  $V$  represents the set of all stations and set  $E$  contains an edge  $(u, v)$  if  $u$  can directly communicate with  $v$ . The network-level neighbourhood of a station  $s$  is then defined as the set of stations within  $N$  hops from the station  $s$ . We assume direct communication only between geographical neighbours.

## 2.3 Event Semantics

As described in [4] each station picks up radio signals with an antenna. These signals are digitized and filtered locally. The filtered signal is analysed for pulses above a certain threshold, which may indicate the occurrence of a cosmic ray. Such an event generates what is called an **N1 trigger**. In fact, the N1 trigger is equivalent to what is called the “level 2” trigger in [4]. Each trigger is timestamped at nanosecond accuracy. For each trigger, in addition to the timestamp, a digitized portion of the signal of 12.5 kilobytes is also buffered at the station. This buffered data along with the timestamp is sent to the CRS upon positive decision through a data analysis procedure; otherwise both the timestamp and buffered data are ignored.

The triggers of two geographically neighbouring stations are said to be *coincident* if their timestamp difference  $\Delta T$  is less than  $T_c$ , the travel time of light in a straight line from one station to the other station. An N1 trigger at a station promotes to an **N3 trigger** in two cases:

- (1) the N1 trigger at a station is coincident with N1 triggers of at least two other geographical neighbours
- (2) the N1 trigger at a station is coincident with an N3 trigger of any of its geographical neighbours.

Normally, an N1 trigger at a station is discarded if it does not promote to an N3 trigger. However, due to link failures, it may not be possible to safely discard data, in which case the N1 trigger is still sent to the CRS.

Figure 1 illustrates the occurrence of two independent cosmic ray air shower events. The event regions are shown shaded and labelled as  $R1$  and  $R2$ . Each station in the event

region has an N1 trigger. For illustration, we assume that the triggers in an event region are coincident. The assumption holds for both  $R_1$  and  $R_2$ . We have the following two cases:

- **General case:** Each of the stations  $A$ ,  $B$ , and  $C$  or any other station in  $R_1$  can be in coincidence with at least two other geographical neighbours in  $R_1$  and promotes its N1 trigger to an N3 trigger.
- **Special Case:** Station  $F$ , as opposed to stations  $D$  and  $E$  or any other station in  $R_2$  has only one geographical neighbour in the event region. Apparently, the N1 trigger of station  $F$  cannot find coincidence with two other neighbors in  $R_2$ . However, station  $F$  is required to promote its N1 trigger to an N3 trigger. Any cosmic-ray detection technique must be able to handle this special case besides the general case.

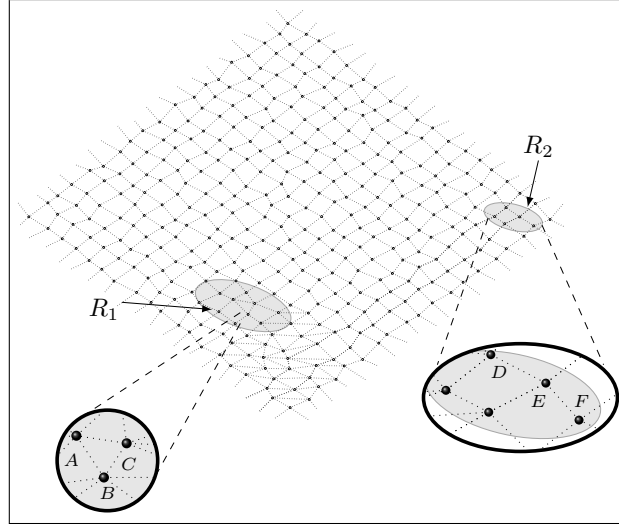


Figure 1: Two independent cosmic-ray air shower events. The event regions are shown shaded. Stations  $A$ ,  $B$ , and  $C$  illustrate a general case in which their N1 triggers promote to N3 triggers. Stations  $D$ ,  $E$ , and  $F$  illustrate a special case of promoting the N1 trigger of station  $F$  to an N3 trigger. Note that station  $F$  has only one neighbour in the event region.

Figure 2 is a sample skymap of measured N3 triggers [4]. It shows the sky in polar coordinates, where the center is directly overhead, and the dark circle near the edge is the horizon. The  $z$ -axis (color scale) is the N3 trigger density in  $\log(\text{eventdensity}/\text{a.u.})$ . It is clear from Figure 2 that the dominant number of N3 triggers are caused by several point sources from the horizon. There are relatively fewer N3 triggers that indicate the direction of arrival of cosmic rays. This implies the need for criteria to filter out N3 triggers that are caused by point sources from the horizon. To carry out such filtering, we introduce a sense of direction.

The direction of the signal that caused the N3 trigger is reconstructed using timestamps and geographical positions of the stations that took part in the coincidence. The direction reconstruction uses what is known as a *plane wave fit*. The reconstructed

direction is represented as a tuple of *zenith* and *azimuth* angles. An N3 trigger becomes an *event of interest* if the direction of signal causing the N3 trigger is within a user-defined range of directions representing the horizon.

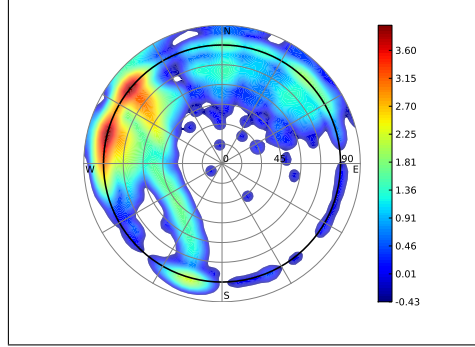


Figure 2: A polar skymap of the reconstructed direction of sample N3 triggers, showing several man-made pulse radio sources on the horizon. The color scale indicates  $\log(\text{eventdensity}/\text{a.u.})$ .

The direction reconstruction process may fail for various reasons. First, the timestamps of N3 triggers may not be all from the same (real) signal. For example, 1 or 2 timestamps may be from accidental coincidences, and in this case there is no unique direction. Second, the timestamps are all from a real signal, but considering that the direction reconstruction process uses heuristics to compute direction, it may not converge and compute an incorrect result. In either case of direction reconstruction failure, the N3 trigger is considered as a *false positive*. In case of a false positive, the timestamp and associated buffered data of the N3 trigger is sent to the CRS. Figure 3 summarizes all possible state transitions of an N1 trigger.

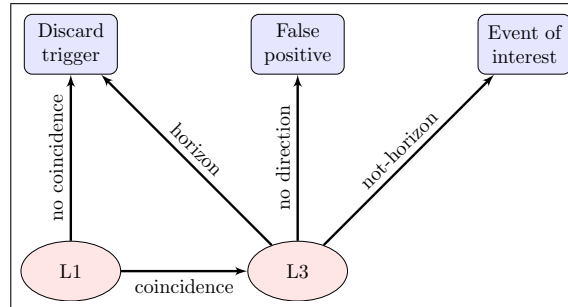


Figure 3: State transition diagram of N1 trigger.

## 2.4 Motivation for Collaborative Local Data Analysis

A challenge of the radio detection of cosmic rays is that the antennas sense not only radio pulses emitted from air showers but are triggered also by pulsed random noise and man-made disturbances (e.g., power transformers or airplanes). While several techniques have been developed to distinguish air shower pulses from man-made

noise (see Ref. [4]), the system must still be robust to elevated and highly variable trigger rates.

A typical initial trigger rate at an AERA antenna is roughly 200 Hz. Since each trigger has an associated 12.5 kB of data, this means that every detector must, in principle, relay a huge amount of data to the CRS over wireless links, which is practically impossible. This is the primary motivation for a hierarchical trigger scheme. Requiring at least 3 antennas in time coincidence can reduce the rate to around 20 Hz.

As shown in Figure 2, around 90% of these triggers can be localized at the horizon, and are therefore likely man-made noise. However, a directional reconstruction requires timing information from multiple antennas. In a centralized system, this decision must be made at a single point that collects all the trigger times from all of the stations.

Collaborative local data analysis can play a pivotal role to discard the uninteresting man-made noise events. In principle, each station is able to find a time coincidence with its geographical neighbours, if it knows their timestamps. A station can then decide locally whether its **N1** trigger is an event of interest. This idea is the main motivation behind devising a distributed event detection algorithm that will allow to build scalable and energy efficient solutions for ultra-high-energy cosmic-ray detection.

### 3 The Distributed Event Detection Algorithm

The goal of our Distributed Event Detection (DED) algorithm is twofold. Firstly, to decide whether an N1 trigger generated locally at a station should be promoted to an N3 trigger. Secondly, provided that an N1 trigger has been promoted to an N3 trigger, to reconstruct, locally, the direction of the radio signal that caused the trigger. Recall that the direction reconstruction helps filter out man-made disturbances including signals transmitted by point sources from the horizon. The direction is reconstructed using timestamps and positions of the neighbouring stations that helped to promote the N1 trigger to an N3 trigger.

#### 3.1 Algorithm I: A Conceptual View of DED

As a first attempt, we design a distributed algorithm that takes decisions based on local information. A station can communicate with all of its geographical neighbours, and we assume no communication failure.

The basic idea of the distributed event detection is the following. Whenever an N1 trigger occurs at a station, the station stores the N1 trigger locally and informs all of its geographical neighbours. The information consists of a message of type **N1Entry** whose structure is shown in Figure 4. Furthermore, when a station receives N1 triggers from its neighbours, it looks for a coincidence of the received triggers with its local ones. A station promotes its N1 trigger to an N3 trigger if its N1 trigger has coincidence with N1 triggers of at least two geographical neighbours.

To cover stations on the boundary of an event region with only one geographical neighbour in the event region, a station not only requires to broadcast its N1 triggers, but also its N3 triggers. We call the latter type of broadcast messages **advertisements**. The information contained in an advertisement are of type **AdvertEntry** whose structure is shown in Figure 4. An **AdvertEntry** entry contains (1) the timestamp of the local N1 trigger that was promoted to an N3 trigger, and (2) the two N1 triggers that had coincidence with the local N1 trigger. An advertisement helps a station promote its N1

trigger to N3 trigger if its N1 trigger has coincidence with the N3 trigger contained in the advertisement message.

```

type N1Entry = record
    srcId : integer;
    seconds : integer;
    nanoSeconds : integer;
end;{ type N1Entry }

type AdvertEntry = record
    srcId : integer;
    seconds : integer;
    nanoSeconds : integer;
    NeighbourN1 : N1Entry[];
end;{ type AdvertEntry }

```

Figure 4: Data types for Algorithm I.

```

/** On Local N1 Trigger */
// Runs when a local N1 trigger occurs at station p
localCache.add(N1(p))
for all q ∈ Neighp do
    send < N1(p), q >

/** Receive from Neighbour */
// Runs when receiving an N1 trigger or Advert
receive < N1(q), q >
OR
receive < advert(q), q >

for any qi, qj ∈ Neighp do
    if coincidence(N1(p), N1(qi), N1(qj)) then
        N1(p) → N3(p)
        process(N3(p))
        for all q ∈ Neighp do
            send < advert(p), q >
        localCache.remove(N1(p))

for any q ∈ Neighp do
    if coincidence(N1(p), advert(q)) then
        N1(p) → N3(p)
        process(N3(p))
        localCache.remove(N1(p))

/** On Remove Trigger */
// Runs when a local N1 trigger is marked for
// removal under the cache eviction policy
if NOT isDecided(N1(p)) then
    apply user defined criteria

```

Figure 5: Pseudocode for Algorithm I.

Figure 5 shows the pseudocode of our algorithm. When an N1 trigger occurs at a station  $p$ , it adds the trigger to its local cache. The local cache has a limited capacity. On the other hand, there is continuous stream of new triggers arriving into the cache due to which the cache may become full thus requiring a *cache eviction* policy. To this end, we consider the cache as a FIFO queue and remove the oldest trigger when the cache becomes full. Therefore, the algorithm is required to process each trigger before it is removed under the cache eviction policy. This requirement imposes a time



constraint on the processing of each trigger. Obviously, the time span with in which a trigger should be processed by the algorithm depends on the size of the cache. Next, the station also informs all of its geographical neighbours about the occurrence of its N1 trigger by broadcasting a message of type N1Entry.

The station then starts listening to (1) N1 triggers from any of its geographical neighbours, and (2) advertisement messages from any of its geographical neighbours. Whenever an N1 trigger is promoted to an N3 trigger, it will be discarded by the station after a computation that involves the following:

- Direction reconstruction of the signal that caused the trigger.
- Deciding, based on the reconstructed direction, if the trigger is an *event of interest*, *not an event of interest*, or a *false positive*.

In case that station  $p$  promotes its N1 trigger to an N3 trigger because of coincidence with N1 triggers of two of its geographical neighbours, the station informs all its geographical neighbours by broadcasting a message of type AdvertEntry. On the other hand, if station  $p$  promotes its N1 trigger to an N3 trigger because of coincidence with an N3 trigger of any of its geographical neighbours, the promoted N3 trigger is not broadcast to the neighbours. The reason for this is that station  $p$  has only one neighbour in the event region which has already promoted its corresponding N1 trigger to N3. For station  $p$ , broadcasting the advertisement in this case is useless. Once an N1 trigger is promoted to an N3 trigger it is removed from storage, in order to be sent to the CRS.

When a trigger is marked for removal under the cache eviction policy and it has not yet been decided by the algorithm (if the trigger is an N3) then there are two possibilities to decide about this trigger before being removed. First, under the assumption of reliable communication, the trigger is discarded; concluding it is not an N3 trigger. Second, under the assumption of unreliable communication, which we do not consider in this paper, station  $p$  may apply some user-defined policy. The user-defined policy is usually a heuristic filter that is applied to the local N1 triggers of a station. One such heuristic filter is that the N1 triggers occur at repeated time intervals. For example, some N1 triggers are caused by electrical power lines passing over the field where antenna stations are installed. These triggers show periodicity correlating with the usual AC power line frequency of 50 or 60 Hz and should be ignored.

### 3.2 Algorithm II: A Periodic Broadcast Algorithm for DED

We notice that each station broadcasts its local N1 triggers at a rate of 200 Hz next to broadcasting its advertisement messages. Our previous algorithm suffers from high bandwidth consumption. To reduce bandwidth consumption, we use an alternative algorithm discussed in this section.

The idea behind this algorithm is that we can save significant amount of bandwidth by grouping local N1 triggers that share the seconds field. The triggers are bundled such that N1 triggers with different values of nanoseconds field share a common value of the seconds field.

Figure 6 compares messages used in both algorithms to broadcast four N1 triggers generated at a station to its geographical neighbours. We assume that the timestamps of all the four triggers have the same value for the seconds field. It is obvious that our first algorithm uses one message per N1 trigger and broadcasts an extensive amount

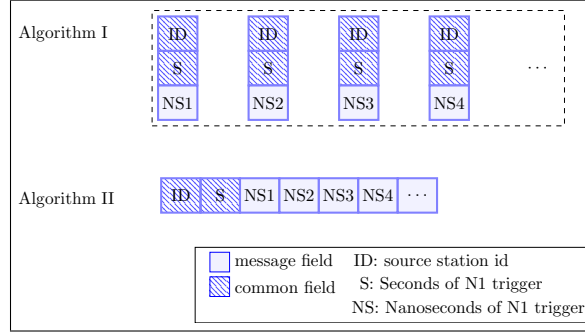


Figure 6: Grouping N1 triggers having the same value of seconds field into a bundle.

of redundant information in the form of the `srcId` and the `seconds` fields. Our modified algorithm can save significant amount of bandwidth, without loss of accuracy, by simply grouping together the N1 triggers having the same value for the `seconds` field. We call the N1 triggers grouped in this way an **N1 bundle**, as shown in Figure 7. In our system, N1 triggers occur at each station at an average rate of 200Hz. We see that by *bundling* these N1 triggers together, having the same value for the `srcId` field and assuming they share the same value for the `seconds` field, the bundle will carry only one instance of both `srcId` and `seconds` fields and a list of distinct nanoseconds. This reduces, in principle, the bandwidth consumption by around 66%.

Similar to the N1 bundle formation, the local N3 triggers that share the same value for the `seconds` field can be bundled as an **advertisement bundle**, also shown in Figure 7. The bandwidth consumption can further be reduced by compressing bundles before their broadcast. Note, however, that this involves computational overhead caused by compression and decompression.

```

type N1Bundle = record
  srcId : integer;
  seconds : integer;
  nanoSeconds : integer[];
end;{ type N1Bundle }

type N1Entry = record
  srcId : integer;
  seconds : integer;
  nanoSeconds : integer;
end;{ type N1Entry }

type AdvertBundle = record
  srcId : integer;
  seconds : integer;
  advertBundleEntry
    : AdvertBundleEntry[];
end;{ type AdvertBundle }

type AdvertBundleEntry = record
  nanoSeconds : integer;
  neighborN1 : N1Entry[];
end;{ type AdvertBundleEntry }

```

Figure 7: Data types for Algorithm II.

Figure 8 shows the pseudocode of our revised algorithm. When an N1 trigger occurs at a station  $p$ , it adds the trigger to its local cache. Due to limited storage capacity each trigger is discarded from the local cache under a certain *cache eviction* policy. We use a simple cache eviction policy that removes the oldest trigger from the cache. The trigger is also added to a local N1 bundle that will be broadcast to geographical neighbours of the station.

The algorithm executes two threads: active and passive. The active thread is executed periodically. It broadcasts N1 bundles and advertisement bundles of a station

```

/** On Local N1 Trigger */
// Runs when a local N1 trigger occurs at station p
localCache.add(N1(p))
N1Bundle.add(N1(p))

/** Active thread */
// Runs every T seconds
for all q ∈ Neighp do
    send < N1Bundle(p), q >
for all q ∈ Neighp do
    send < AdvertBundle(p), q >

/** Passive thread */
// Runs when receiving an N1Bundle or AdvertBundle
receive < N1Bundle(q), q >
OR
receive < advertBundle(q), q >

for all entry ∈ Bundle do
    coincidence = localCache.coincidence(entry)
    if coincidence then
        N1(p) → N3(p)
        process(N3(p))
        if entry ∈ N1Bundle do
            N1Bundle.add(entry)
        localCache.remove(N1(p))

/** On Remove Trigger */
// Runs when a local N1 trigger is marked for
// removal under the cache eviction policy
if NOT isDecided(N1(p)) then
    apply user defined criteria

```

Figure 8: Pseudocode for Algorithm II.

to the geographical neighbours of the station. The passive thread listens to incoming messages. Upon receipt of an N1 bundle or an advertisement bundle from a geographical neighbour, the thread looks for a coincidence of each trigger in the bundle with the local N1 triggers. Whenever an N1 trigger is promoted to an N3 trigger, it will be discarded if it came from an invalid direction as before.

If a coincidence has been found among the local N1 trigger and N1 triggers of the geographical neighbours, then these N1 triggers are added to the advertisement bundle.

When a trigger is marked for removal under the cache eviction policy and it has not yet been decided by the algorithm (if the trigger is an N3) then it is treated exactly the same way as before.

## 4 Design Space Analysis

We measure the accuracy of our algorithms according to the number of false negatives and efficiency according to the number of false positives and the amount of communication among geographical neighbours during event detection. Ideally, the algorithms should neither produce false negatives, nor false positives. However, due

to the constraints of limited amount of communication bandwidth, storage, and computational power, we need to consider tradeoffs that cause false negatives and false positives. These tradeoffs are dictated by the communication overhead, storage requirements, and computational complexity.

	Local Comput.	Local Storage	Local Comm.	CRS Comm.	False negatives	False positives
1. Use bundles	+	+	-	o	o	o
2. Compress bundles	+	+	-	o	o	o
3. Apply Bloom filters	+	+	-	+	o	+
4. Encode local N1 triggers	+	-	o	o	o	o
5. Increase broadcast frequency	+	-	+	o	o	o
6. Increase neighbourhood	+	+	+	+	o	+
7. Use domain-specific filters	+	+	o	-	+	-

+ increase  
 - decrease  
 o no effect

Figure 9: Design space exploration for collaborative local data analysis techniques.

We use our proposed first algorithm under ideal conditions as reference point for comparison. There are several cases related to our revised algorithm where tradeoffs can be considered. These cases are discussed below and summarized in Figure 4.

1. By using bundles, we reduce local communication at the price of increased local computation and (temporary) storage.
2. Compressing bundles will further decrease communication costs, yet also increase local computational effort. In addition, more local storage is needed in comparison to not using any bundles at all, although compression will help to keep this required additional storage low.
3. Probabilistic data structures like Bloom filters [5] for exchanging N1 triggers, result in reduced communication during local data analysis. The lookup operation is cheaper. However, this technique increases the number of false positives, in turn, implying increased communication overhead with the CRS.
4. We can reduce the memory usage at the cost of increased computation as follows. Instead of storing the timestamp of a local N1 trigger in a pair of seconds and nanoseconds fields, we store it as an offset to a certain base time. The full information on an N1 trigger can be computed back using this base time and the offset.
5. If the frequency with which a station broadcasts its N1 bundle and advertisement bundle is increased, then it will allow stations to decide on their local N1 triggers earlier. Since an N1 trigger is removed from the local cache when it has been evaluated, increasing the broadcast frequency will result in reduction of memory usage for storing N1 triggers. On the other hand, the increased broadcast frequency will increase the number of bundles exchanged overall, and this

increased number of bundles will increase the computational overhead required for each bundle. Moreover, an increased broadcast frequency will also consume more bandwidth which means consuming more energy.

6. For the same transmission range, increasing the neighbourhood size beyond a certain minimum has a negative effect on performance. The reason is that more N1 triggers are generated whose collaborative processing consumes resources for no gain. Note that this statement is based on the assumption of reliable communication channels. In case of unreliable communication channels, increased neighbourhood size will improve the robustness of the system.
7. As mentioned earlier, a station may apply some user-defined heuristic filter to the local N1 triggers of the station. The heuristic filter discards those N1 triggers which fall within certain time windows along the time domain. If the length of the window is kept too small then fewer N1 triggers will be discarded. On the other hand, keeping a larger window the filter may discard some N1 triggers which are potential N3 triggers. This situation gives rise to false negatives.

## 5 Performance Evaluation

### 5.1 Experimental Setup

We carried out simulations to demonstrate the accuracy and efficiency of our distributed event detection algorithms. The simulations were conducted using the OM-NET++ simulation environment. We used traces of N1 triggers collected from a small-scale testbed for cosmic-ray detection [4]. The testbed consists of 22 stations and uses wired infrastructure for communication between stations and the CRS. The data analysis procedure in the testbed is centralized: every station sends its N1 triggers to the CRS for analysis. It is important to emphasize that the occurrence of N1 triggers is independent of the data analysis procedure. So the N1 triggers generated in the testbed with wired infrastructure and centralized data analysis procedure could still be used to validate and analyse our proposed solution based on wireless infrastructure and collaborative local data analysis procedure.

To validate our approach, we compared the functionality of our distributed algorithms to that of the centralized approach. We simulated the behaviour of our centralized approach by analysing the real-world traces containing N1 triggers and computing N3 triggers for each station. Next, we simulated our algorithms using those enabling stations to compute N3 triggers locally. The N3 triggers computed for a station through our centralized approach should be the same as computed through our distributed event detection algorithms.

In order to analyse the effect of using collaborative local data analysis, we considered a representative station  $S$  in our simulation. Recall that our centralized approach would have sent all the N1 triggers generated at station  $S$  to the CRS. We were interested to see whether station  $S$ , under our collaborative local data analysis approach, can actually reduce the data that is sent to the CRS. To that end, we monitored the frequency of the following variables for station  $S$ :

- N1 triggers
- N3 triggers

- False positives
- Events of interest

Here, an event of interest is defined as the N3 trigger for which the direction reconstruction process is successful and the *zenith* angle (in degrees) lies outside the interval  $[85, 95]$ . The signal arriving from an angle within the interval  $[85, 95]$  is considered as a signal from the horizon (i.e. man-made disturbance).

As argued in Section 3.2, our enhanced algorithm can save a significant amount of bandwidth by bundling consecutive local N1 triggers before broadcasting to geographical neighbours. To see this effect we compared the bandwidth utilization of our basic algorithm and our enhanced algorithm with respect to the following variables for station  $S$ :

- Transmission rate (bits/sec)
- Reception rate (bits/sec)

Finally, we examined the effect of additional computation at the stations on communication. To that end, we applied compression to the N1 bundles and advertisement bundles in our enhanced algorithm. We used lossless compression (*zlib-1.2.5*) so that the receiving station is able to reconstruct the original bundle.

## 5.2 Results

We first focused on validation of our collaborative local data analysis approach. By inspection of the actual real-world traces, we verified that each node indeed observed the same N3 triggers that should have been noticed. This means that our algorithms are functioning correctly.

Let us now consider the local filtering capability of our algorithms, by focusing on our representative station  $S$ . Figure 10 shows the local data analysis performed at

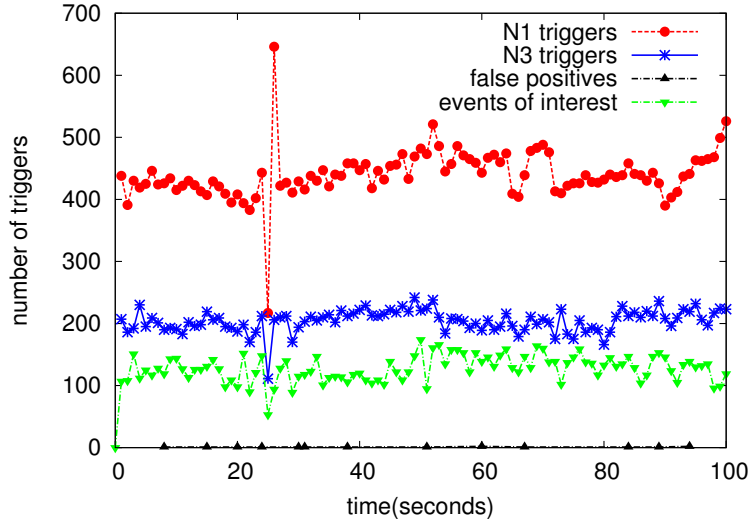


Figure 10: Collaborative local data analysis at a station  $S$ .

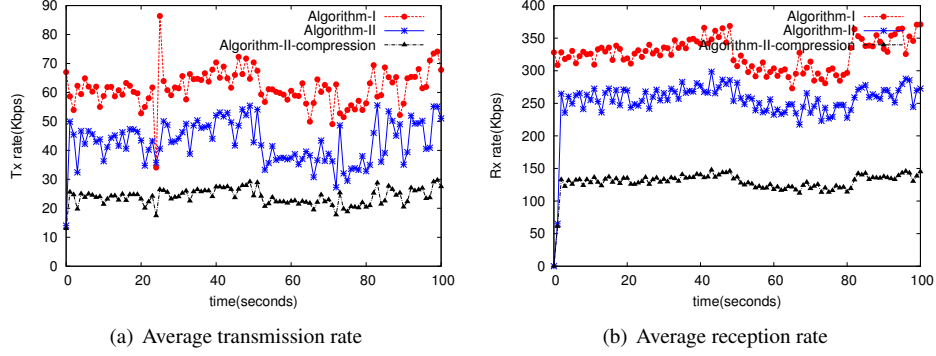


Figure 11: Performance comparison of Algorithm-I and Algorithm-II with respect to communication bandwidth consumption for station  $S$ .

station  $S$  over a period of 100 seconds. The frequency of N1 triggers (generated at station  $S$ ) during the experiment is shown. Next, as a result of executing our distributed event detection algorithms, the fraction of N1 triggers at station  $S$  that were promoted to N3 triggers is shown. For each N3 trigger, the algorithm attempts to reconstruct the direction of the signal that caused the N3 trigger. By definition, an N3 trigger becomes a *false positive* if the direction reconstruction process for that trigger fails. Figure 10 shows the fraction of N3 triggers that were marked as false positives by our algorithms. Finally, the fraction of N3 triggers is shown for which the direction reconstruction process was successful and each trigger fulfilled the condition of an event of interest.

Wireless communication offers limited bandwidth, therefore, efficient utilization of bandwidth becomes essential. From Figure 10 we can see that our distributed algorithms enable a station to discard a huge amount of data by first communicating only with its geographical neighbours. Only a relatively small amount of data is sent to the CRS for further analysis. This is a positive indication for efficient bandwidth utilization.

We see in Figure 10 that a small number of false positives is generated by our algorithms. In our sample trace of the N1 triggers for station  $S$ , we notice that the rate of false positives do not exceed two triggers per second. A *false positive* is a consequence of the direction reconstruction process failure. As discussed in Section 2, there are two reasons for this failure. First, there might be no real signal underlying the corresponding N3 trigger. If we were sure that there is no real signal underlying the N3 trigger, then instead of reporting it as a false positive we could confidently discard it locally. This would enable the station to send even less data to the CRS. Second, the direction reconstruction process, using heuristics, may fail to reconstruct direction for an N3 trigger caused by a real signal. In both cases we need more efficient methods for direction reconstruction which will help in further reduction of the amount of data sent to the CRS.

Efficient utilization of bandwidth, under high frequency of N1 triggers, also involves several tradeoffs. As an example we consider our definition of **event of interest**, which we defined to be an N3 trigger whose zenith angle lies outside the interval  $[85, 95]$ . Note that the interval represents the direction of signals from the horizon. As we increase this interval, more N3 triggers will be discarded. This means that the amount of data that is sent to the CRS is further reduced and subsequently more effi-

cient bandwidth utilization. On the other hand, keeping a large interval to represent the horizon may lead to producing *false negatives*: discarding N3 triggers that were caused by an actual cosmic ray air shower. This situation shows a tradeoff between *accuracy* and *communication*.

It is important to note that both of our algorithms produce the same output in terms of N1 triggers promoted to N3 triggers, false positives produced, and N3 triggers declared to be events of interest. The only difference between the two algorithms is in bandwidth utilization. The enhanced algorithm is expected to utilize the bandwidth more efficiently than the basic algorithm. Figure 11 shows a comparison of bandwidth utilization by our basic algorithm, our enhanced algorithm, and our enhanced algorithm with bundle compression. The measurements were performed at our representative station *S*. We see that there is a substantial difference between the bandwidth utilization of the basic algorithm and the enhanced version.

Figure 11 also demonstrates the effect of compressing N1 bundles and advertisement bundles in our enhanced algorithm before broadcasting to its geographical neighbours. The bundle compression significantly reduces the number of transmitted bits per second. The effect of compression becomes more pronounced in case of reception of compressed bundles. This was to be expected considering the fact that station *S* receives from multiple neighbours. We believe the effect of compression can be made more significant by applying compression techniques that consider the nature of the data we are handling.

## 6 Related Work

The common model for event detection in wireless sensor networks (WSNs) is that each node relays all of its locally generated data to the base station without local processing [6]. This model works well for small-scale networks, a small amount of data per event, and lower rates of events per node. This model is inefficient for large-scale networks, as aggregated data transmissions can easily exceed the available bandwidth en route to the central station.

Another model for event detection involves in-network processing. In this model, processing is done by the nodes to compute events of interest against certain criteria known to the nodes. This may significantly reduce the amount of communication and, hence, the energy consumed.

In TAG [7] data is processed along the routing path at its intermediate nodes. This approach works well for computing aggregates like *max*, *min*, *count*, and *sum* etc. However, it is not suitable for event-detection schemes where acknowledging the node about detection of event of interest is mandatory.

In [8], the network is divided into equally sized cells. Each cell has a leader. Nodes within the cell route their data to a leader. The leader processes the data and informs other nodes in the cell about the decision. This scheme is scalable and also satisfies the requirement of acknowledging node about the decision. However, this scheme will easily produce false negatives in case an event occurs on the border of two or more cells.

In [9], Wittenburg et al. present a distributed event detection scheme where each node decides itself based on information from its neighbours. Their scheme scales well. The node is also aware of the final decision made about the occurrence of an event. However, they do not explore the tradeoffs associated with this scheme under a high frequency of local events.



In [10], Werner-Allen et al. deal with handling high rates of events up to 102 Hz per node. They use a distributed event detection scheme that works as follows. When a node triggers a local event, it broadcasts a vote message. If any node receives enough votes from other nodes during some time window, it initiates global data collection by flooding a message to all nodes in the network. This scheme implies that when an event occurs it is detected by *all* nodes in the network. However, based on this assumption, the scheme is not scalable.

Finally, Werner-Allen et al. argue in [11], to use a WSN as a scientific instrument. They deal with an event rate of 100Hz per node and high resolution data. Because of the high data rates it is infeasible to transmit all sensor data. So nodes locally detect interesting events and only transmit data related to interesting events. They use an event detection algorithm that is basically centralized. When a node triggers an event, it is transmitted to the base station. If the base station receives triggers from 30% of the active nodes within a 10 second window, then it is considered as an event of interest and data collection is initiated. However, this scheme of event detection is not scalable for large geographical areas and only applicable to specific domains, with network-wide events.

Concluding, the requirements that we mentioned have been partially addressed by a multitude of schemes. However, to the best of our knowledge, no work lies in the intersection of these areas to propose a scalable solution and address the design trade-offs for the applications class we target at under the constraints of limited energy and capacity of the WSN nodes.

## 7 Conclusion and Future Work

We notice that our proposed distributed event detection algorithms use information only from geographical neighbours and perform analysis locally at the station to discard irrelevant data. In contrast, a centralized approach requires each station to send its data (N1 triggers) to the CRS for analysis. This implies that as the number of hops between the station and the CRS increases, the communication cost of sending the N1 triggers to the CRS also increases. This means that the centralized approach is not geographically scalable. On the other hand, our algorithms enable a station to analyse data locally and show strong affinity for geographical scalability.

In this paper we focused on exploring the possibility of collaborative local data analysis to build geographically scalable solutions for ultra-high energy cosmic ray detection. The devices used for *cosmic ray air shower* detection have limited communication, processing, and storage capacity. Each detector also has a limited energy budget.

Wireless communication among cosmic-ray detectors is the only way to cover a large spatial area for detecting cosmic-ray air showers. Each cosmic-ray detector generates a huge amount of data that needs to be sent to the CRS. But wireless communication medium offers limited bandwidth that does not meet the bandwidth requirements of a centralized solution for this application.

We present a distributed event detection algorithm that enables a cosmic ray detector to analyse its data locally based on information from its geographical neighbours. The ability of collaborative local data analysis makes our algorithm attractive for building large geographically scalable solutions. The results from simulating our algorithm show that a significant amount of irrelevant data can be filtered out locally. This allows a detector to utilize wireless communication bandwidth more efficiently. We also ex-

plore several design tradeoffs that help understand the relationship between accuracy of our algorithm and various resources usage.

We know that wireless communication medium has not only limited amount of bandwidth but wireless communication links are also unreliable. In this paper, we assume that links are reliable. As future work we will explore the behaviour of our distributed event detection algorithm under unreliable communication channels.

## 8 Acknowledgements

We thank the Pierre Auger Collaboration for using part of their data sets and stimulating discussions with C. Timmermans and A.M. van den Berg in an early stage of the project.

## References

- [1] S. Yousaf, R. Bakhshi, and M. van Steen. Reliable localized event detection in a wireless distributed radio telescope. In *Proc. Workshop on Sensor Networks in conjunction with ICCCN 2012*, pages 1–6. IEEE, 2012.
- [2] R. Clay et al. Prospects for  $10^{19}$  eV cosmic ray studies in South Australia. *Nuclear Physics B - Proceedings Supplements*, 28(2):101 – 122, 1992.
- [3] J. Abraham et al. (Pierre Auger Collaboration). Properties and performance of the prototype instrument for the Pierre Auger Observatory. *Nucl. Instrum. Meth.*, A523:50–95, 2004.
- [4] J. L. Kelley for the Pierre Auger Collaboration. Data Acquisition, Triggering, and Filtering at the Auger Engineering Radio Array. *Proc. VLVnT11 Workshop*, arXiv:1205.2104, 2012.
- [5] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, pages 422–426, 1970.
- [6] J. Gehrke and S. Madden. Query processing in sensor networks. *IEEE Pervasive Computing*, 3:46–55, 2004.
- [7] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: a Tiny AGgregation service for ad-hoc sensor networks. *SIGOPS Oper. Syst. Rev.*, 36:131–146, December 2002.
- [8] F. Martincic and L. Schwiebert. Distributed Event Detection in Sensor Networks. In *Proc. Conf. Systems and Networks Communication (ICSNC)*, pages 43–48. IEEE Computer Society, October 2006.
- [9] G. Wittenburg, N. Dziengel, C. Wartenburger, and J. Schiller. A system for distributed event detection in wireless sensor networks. In *Proc. Conf. Information Processing in Sensor Networks*, pages 94–104. ACM, 2010.
- [10] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh. Monitoring volcanic eruptions with a wireless sensor network. In *Proc. European Workshop on Wireless Sensor Networks (EWSN)*, pages 108–120. IEEE, 2005.

- [11] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh. Fidelity and yield in a volcano monitoring sensor network. In *Proc. Symp. on Operating systems design and implementation (OSDI)*, pages 381–396. USENIX Association, 2006.